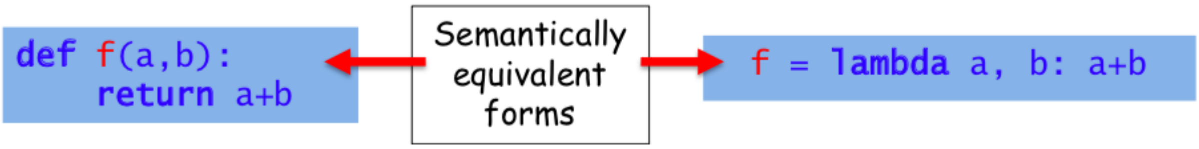
[Lecture 5] (Even) More on functions

How to call a function:

1. `expression(arguments)`
2. `f = cube`

   `f(3)`
3. `(sqr if cond else cube)(3)`

Lambda functions - unnamed functions



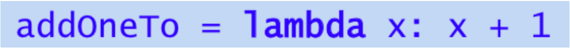
```
def f(a,b):
    return a+b
```

Semantically equivalent forms

```
f = lambda a, b: a+b
```

Lambda depends on the **local** context

Closed term

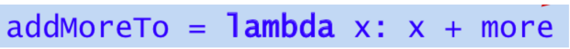


```
addOneTo = lambda x: x + 1
```

x - **bound variable**

Open term

`more = 1000`



```
addMoreTo = lambda x: x + more
```

more - **free variable**(bounds to lambda dynamically)

`addMoreTo(5)` # returns 1005

`addMoreTo(5)` # ERROR more is not defined

`more = 1`

`addMore = lambda x: x + more`

If more changes after lambda was created?

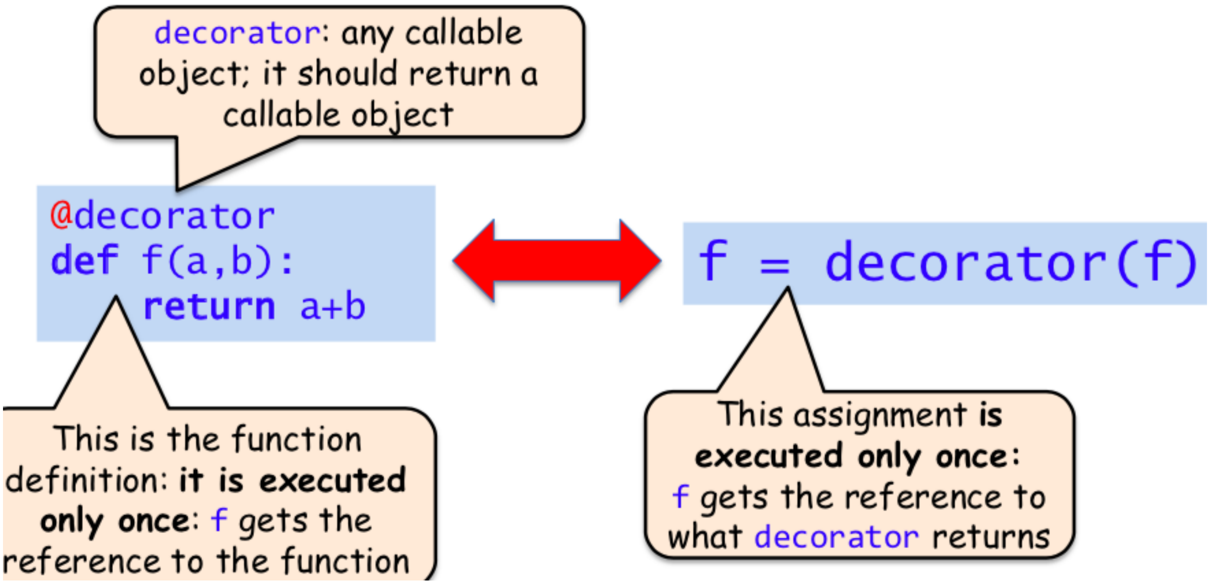
`more = 99`
`addMore(1)`

Returns 2 if bound by ***value***,

`= 100` lambda captures the **reference** to the free variable in

or 100 if bound by ***reference***	Python

Function **Decorators**



```
@decorator
def f(a,b):
    return a+b
```

```
f = decorator(f)
```

decorator: any callable object; it should return a callable object
This is the function definition: it is executed only once: f gets the reference to the function
This assignment is executed only once: f gets the reference to what decorator returns

Class **Decorators** - adding some extra functionality and the state

function is **stateless** (no direct way to pass information from one function call to another)

class has **state** (attributes) and **behaviour** (methods)

`f = decorator(f)` ⇒ f is a class instance

`f(1, 2)` == `f.__call__(1,2)` - special instance method that is called for the construct of the form class-instance(arguments)

---

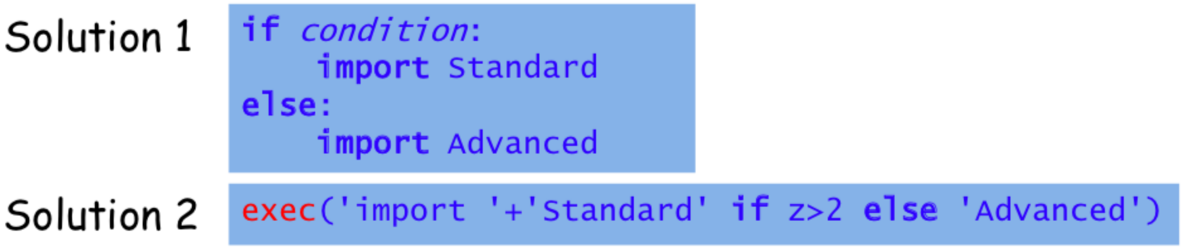
eval

- Accepts a string (+optional context params)

- Assumes that the string contains a text that satisfies the syntax of an **expression**
- Parses it…
- …**And evaluates it!**

exec

- Accepts a string (+optional context params)
- Assumes that the string contains a text that satisfies the syntax of a **statement**
- Parses it…
- …**And evaluates it!**



Solution 1

```
if condition:
    import Standard
else:
    import Advanced
```

Solution 2

```
exec('import '+'Standard' if z>2 else 'Advanced')
```